A NoSQL Database Approach for Modeling Heterogeneous and Semi-Structured Information

Gerasimos Vonitsanos*, Andreas Kanavos*,†, Phivos Mylonas*, Spyros Sioutas*,†
*Department of Informatics
Ionian University, Corfu, Hellas
{c16voni, fmylonas, sioutas}@ionio.gr
†Computer Engineering and Informatics Department
University of Patras, Patras, Hellas
kanavos@ceid.upatras.gr

***Abstract*—Nowadays there is a growing need for collecting and processing data from different sources in heterogeneous and semi-structured formats. Scientists and companies are strongly urged to find a way for extracting knowledge out of them. In this paper, we present a NoSQL database approach for modeling heterogeneous and semi-structured information in both software architecture and data modeling aspects. We built a robust analytics framework by integrating Apache Spark with Apache Cassandra and in following utilize data mining techniques for presenting a model capable of predicting the relationship between tourist arrivals and nights spent in Greece. The proposed model puts to use a constructed dataset both from the Hellenic Statistical Authority and Eurostat. The evaluation shows that the proposed data model, used for fitting the current dataset, predicts tourist behaviour with high accuracy.**

***Index Terms*—Apache Cassandra, Apache Spark, Big Data, Cloud Computing, Column-family Stores, Data Modelling, NoSQL Databases**

I. Introduction

The overly increasing amount of data we currently have in our possession, has created the need for their process which will enable the extraction of important knowledge. Organizations are endlessly confronted by more and bigger data challenges having no way to obtain valuable information. The diversity of these sources focuses on the fact that they cannot be processed or analyzed using traditional methods or tools [31]. Therefore, the importance of data analysis affects the organizations providing them with useful knowledge, but at the same time, it has begun to attract the interest of researchers as well.

Cloud computing is based on many years of research in the field of virtualization, distributed computing, software as well as web services. It implements a service-oriented architecture, thus providing reduced information about the used technology to the end user. Besides, it is loaded with great flexibility, reduced total cost and offers many other benefits simultaneously [29].

The outspread of large web applications, which output a vast amount of data that conventional relational databases cannot handle anymore, has led to the rise of a new generation of databases. These databases are entitled NoSQL, short for not only SQL, and principle-wise are not inherently traditional oriented. They can manipulate the outcome of modern large web-scale platforms like Twitter, Facebook, and Google in an efficient way[1]. The data they are dealing with, are not relational and they are superior to traditional databases, which encounter scalability and availability problems due to the data size. NoSQL defines a filter in order to precisely determine the databases that fulfill the aforementioned requirements. Hence, we can conclude that NoSQL databases have a lot more to chip in than merely offer solutions to scale problems [11].

In this paper, we aim to build a robust data analytics platform by integrating Apache Spark alongside Apache Cassandra. Apache Spark is a cluster computing system with a growing ecosystem of libraries and a framework to render advanced data analytics feasible. On the other hand, Apache Cassandra constitutes a NoSQL database for handling large portions of data and provides highly available service with no single point of failure. It also supports a number of astounding characteristics such as replication and multi-data center replication, scalability, fault-tolerance, tunable consistency and query language. We initially construct a dataset using data from the Hellenic Statistical Authority and Eurostat. In following, a data model in order to fit into this particular dataset by using the Cassandra Database, is proposed. Finally, the proposed data model is evaluated through Machine Learning algorithms in a linear regression scenario with use of Apache Spark.

The rest of this paper is organized as follows. Section II presents the related work. Section III presents the open-source cluster-computing framework Apache Spark, the distributed NoSQL database management system, e.g., Apache Cassandra as well as the Machine Learning Algorithm which was utilized in our proposed system. In addition, our proposed model is introduced and further analyzed in Section IV. Furthermore, in Section V, the evaluation of the experiments and the results conducted, is presented. Ultimately, Section VI presents conclusions while Section VII draws directions for future work.

978-1-5386-8161-9/18/$31.00 © 2018 IEEE

[1]https://en.wikipedia.org/wiki/NoSQL

II. Related Work

Big data analytics constitutes the use of advanced techniques in order to extract patterns, trends, as well as other insights from very large, diverse datasets that have one or more of the following characteristics: high volume, high velocity, or high variety.

The data modeling procedure, referring to a collection of objects, has always been a difficult step in the RDBMS systems. All the data need to be stored in different tables and the correct usage of joins is essential to access and aggregate more efficiently the data needed. Of course, the data representation is a massive process for the designers since they have to know the precise database structure [21]. Recent works show early signs of methods often described as best practices on how to successfully migrate from RDBMS to Cassandra. The task is difficult but challenging and requires a proper transformation of the existing data model into a new one utilizing the new features that NoSQL databases provide [3], [8], [9], [17], [19], [26].

NoSQL movement emerged as a way to address the growing need to process and analyze extremely large data volumes as quickly and efficiently as possible and in ways that traditional relational databases couldn't handle. NoSQL provides a flexible structure for working with massive data sets that can be scaled out and distributed across geographically dispersed commodity hardware. This kind of databases are used, for instance, to collect click-stream data from high-volume websites, monitor online retail behavior, aggregate real-time sensor data and support social networking communications.

NoSQL databases are typically categorized according to their organizational structure as it is for the way of storing the data. The only aspect they have in common is the fact that they do not follow a relational data model but this does not provide enough elements about the categorization. However, NoSQL databases are typically classified in the following four categories [11]:

- *Key-Value stores* where data are stored as pairs of attribute names (keys) and values. The application developers can provide the data in a schema-less way because in this type of key-value relationship, the data can be elements of a programming language (integers, arrays) or other objects like files, images, etc. This non-fixed data model is straightforward to be used and can also store and retrieve data at extremely high speeds. The most popular is of course, the famous Amazon Dynamo, but Voldemort and Riak are also well known schemes belonging to this category [28].
- *Column Oriented stores* store data not in rows, as in relational databases, but in columns. The speed of the query results is improved because only the required columns are accessed by the database. The aggregation and data warehousing of the applications is easier to be implemented since data are stored on small blocks instead of tables. Cassandra and HBase are the most popular paradigms of this category.
- *Document stores* store data in files rather than in tables. These files are data structures known as documents and utilize a particular pattern. Each file has a unique key which is named as Universal Unique Identifier (UUID) and is automatically generated. Moreover, the documents can contain encapsulated data with XML, YAML JSON and other types of encodings. In general, the document databases are distinguished by the ability to store big data and to provide high speeds in query results. The most popular databases in this category are MongoDB and CouchDB.
- *Graph Databases* store their data using graph structure instead of tables and columns. The graphs represent the relationships of the data using nodes and edges. The graph structure reduces significantly the need of using complex joins and indexing can be also implemented in a not strict schema. Examples of this category are Neo4j and HyperGraphDB.

The findings in the research works of [10], [20], [27] as well as our proposed work indicate that column oriented systems are more flexible in data modeling than other NoSQL families. In [24], authors propose a method for transforming and migrating data schemas into the column oriented NoSQL database HBase. In [15], a heuristic solution for transforming relational to column oriented NoSQL database is proposed. To the best of our knowledge, there isn't yet a similar work referring to utilizing a data model that fits a dataset and predicts touristic behaviour with high accuracy. Also, this work proposes an operational data modeling methodology for Apache Cassandra in order to implement heterogeneous and semi-structured information originating from different sources of the web.

In our previous studies, we have proposed a cloud-based architecture aiming at creating a sentiment analysis tool for Twitter data, based on Apache Spark cloud framework [6]. There, tweets are classified using supervised learning techniques and the classification algorithms are used for ternary classification. We have also presented a survey of machine learning algorithms, implemented on Apache Spark over DHT-structures (Distributed Hash Table) [27]. These were experimenting across a POS dataset that had been stored in Cassandra with some of the most influential algorithms that have been widely used in the machine learning community. A novel distributed framework implemented in Hadoop as well as in Spark for exploiting the hashtags and emoticons inside a tweet is introduced in [12] and [18]. These works introduce a classification procedure of diverse sentiment types in a parallel and distributed manner. Moreover, Bloom filters are also utilized in order to increase the performance of the proposed algorithm. Finally, one similar work is presented in [2], where it aims to perform classification analysis in a real dataset, using Apache Spark's MLlib. The investigation of several classification models (Decision Trees, Random Forest, Logistic Regression) leads to the way they evolve according to the size of the dataset.

III. Preliminaries

A. Apache Spark

Apache Spark [30] was developed at UC Berkeley's AMPLab and later the Spark codebase was donated to the Apache Software Foundation [2]. The distributed open source processing system is commonly used for big data processing. The fast performance of Apache Spark is due to the fact that it temporarily stores data in memory contributing to optimized execution. The system is suitable for all types of data science, including batch processing, streaming analytics, machine learning, graph databases, etc [3].

Apache Spark, in general, is a high speed optimized engine that offers APIs in Java, Scala, Python and R. It can run standalone or over Hadoop or Mesos and access data sources like HDFS, Cassandra and HBase [4].

The basic engine of the platform, with all the functionality built-in, is called Spark Core. This corresponding module includes components capable of task scheduling, management of memory, fault recovery, interaction with the data storage systems and other. Spark extends the MapReduce model offering high speed because in the previous model all the complex applications were running on disk, while in Spark, the complex computations run in memory [25]. Also, there is no need to maintain separate tools since Spark combines many different processing types [13].

The Apache Spark architecture is illustrated in Figure 1 including Spark SQL and data frame operations. Spark's distributed data-sharing concept is called "Resilient Distributed Datasets" or RDD. RDDs are fault-tolerant collections of objects partitioned across a cluster that can be queried in parallel and used in a variety of workload types. A DataFrame constitutes a collection of distributed row types. These provide a flexible interface and unlike existing data frame APIs in R and Python, DataFrame operations in Spark SQL go through an extensible relational optimizer, called Catalyst. Concretely, Catalyst Optimizer is an extensible query optimization framework. Finally, Spark SQL is a module in Apache Spark that integrates relational processing with Spark's functional programming API. The aim is to query data and in following call complex analytics libraries. Additionally, Spark SQL uses a nested data model based on Hive.

Finally, Spark is built from the ground up exclusively for performance and reliability and takes advantage of the operational and debugging tools developed for the Java stack, since it sits atop the JVM [22].

B. Apache Cassandra

Apache Cassandra is an open source NoSQL database, which is extensively scalable. For this reason, it is ideal for managing huge amounts of data in different data centers and the cloud as well. Some of its characteristics are the provision of availability that is continuous, its scalability that is linear and the simplicity in operating on multiple servers without any single point of failure. The data model of Cassandra is exceptionally flexible and provides swift response times [5].

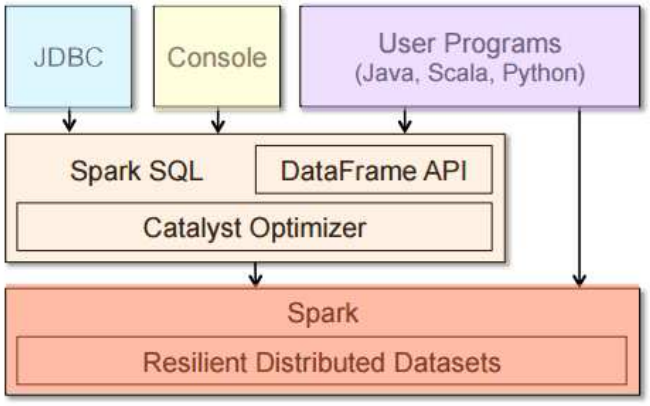

Fig. 1. Apache Spark Architecture

Its design was based on the premise that system/hardware failures always happen and this fact results in a peer to peer distributed system. All nodes are the same, whereas the master node as well as the name nodes are missing. The data are distributed on all cluster nodes and the copying and sharing procedures are automatic and transparent. This method ensures tolerance for any system failures [6].

It uses consistent hashing for assigning data to nodes. Regarding calculation process, the hash function, for the keys of each item that are stored in Cassandra, uses a certain hash algorithm. The range of hash values (otherwise the keyspace) is divided between the cluster nodes. Cassandra then assigns each data to a node, and that node is responsible for storing and managing the given data [7]. Cassandra also provides an advanced custom replication which saves copies of the data on all nodes participating in a Cassandra ring. So if a node is shut down, one or more copies of the information that the node has, will be available to another cluster node. Cassandra provides linear scaling capability which means that the overall capacity of the systems can be quickly increased by adding new nodes to the network.

Unlike relational tables where a column family's schema is not fixed, Cassandra does not force individual rows to have all the columns as presented in Figure 2.

C. Apache MLlib

Spark ships with MLlib[8], a distributed machine learning library [16]. It consists of implementations of common machine learning algorithms, that can be used with scalable environments, for several types of analysis including classification, regression, clustering and collaborative filtering. MLlib also

[2]https://en.wikipedia.org/wiki/Apache_Spark
[3]https://aws.amazon.com/emr/details/spark/
[4]https://spark.apache.org/
[5]https://docs.datastax.com/en/cassandra/3.0/
[6]https://www.datastax.com/resources/tutorials/cassandra-overview
[7]https://www.ibm.com/developerworks/library/os-apache-cassandra/
[8]http://spark.apache.org/mllib/

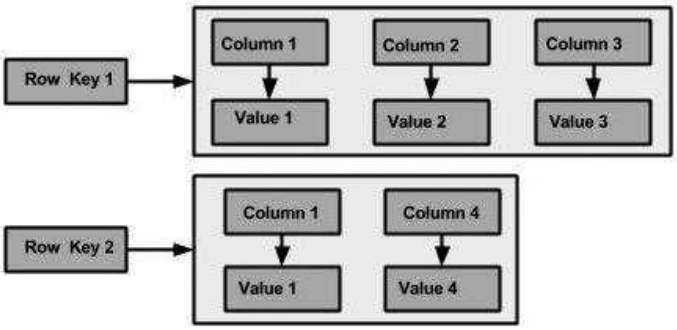

Fig. 2. Apache Cassandra column family

includes Java, Scala and Python APIs. Finally, it is Apache Spark's scalable machine learning library and is developed as part of the Apache Spark Project.

D. Machine Learning Algorithm

The core system of Spark consists of different libraries and components that provide a rich set of higher tools including MLlib for machine learning. This library provides Spark with a framework for machine learning methodologies including multiple types like regression, classification, neural network processing and more. Furthermore, this library is based on the RDD API and can be used with R and Python programming languages. Since Apache Spark can take advantage of multiple computer nodes in a cluster, the library is designed to scale out in a cluster, thus avoiding major memory bottlenecks. Along with MLlib, Apache Spark also includes a DataFrame based machine learning API which is named SparkML, and programmers can choose which library to use according to their dataset and data size to achieve the best performance [14].

In our model, we delve into experimenting with Linear Regression, which is a statistical method for predictive analysis studying the relationship between one dependent variable and one or more independent variables denoted as x [9].

IV. Implementation

A. Model Overview

The absence of joins can be considered as the most significant data modeling difference between distributed and relational databases. Cassandra defines the importance of understanding the implications before starting the data modeling procedure. More specifically, data for a table in distributed databases are spread across all the nodes of the cluster based on the token values that are generated from the partition keys. The data modeling needs to be utilized in such a way that all the data required to answer a query would be available in one table, even if we have to create corresponding tables for each query.

In our proposed methodology, we focus on the application itself and not on the data or their relational theories [5]. The tables are designed having in mind the queries that we need and in following the database schema. Of course in relational databases, our primary concern is to remove data duplication and in this way, to make the database ACID compliant. In Cassandra, the environment is distributed and ACID compliance is difficult to achieve, so the replication of data is required [4].

Therefore, the query modeling method has been used and led us to the creation of four main database tables, containing all the required data. Thus, the creation of these tables gave us the ability to make the required aggregations. Furthermore, a table containing only the data required for the results section was created, complying precisely with Cassandra result oriented model and acting as a pre-built result set. Cassandra does not support complex processing at query time, so the data must be in advance prepared. Since Cassandra is a column-oriented store, which is a multidimensional map, we utilized the map data structure [23]. In this way, we are granted with efficient key lookup and efficient scans. Besides, the extension of the database model with new functionalities is more feasible in the future.

B. Dataset Description

In order to conduct the experiments of a specific case study, a set of data is constructed. This data was collected from the web, and concretely from the websites of the Hellenic Statistical Authority (EL.STAT.) and Eurostat, containing the results of the travel expense border survey from 2006 to 2015. This methodology is appropriate for countries like Greece where the number of entry/exit stations in their territory is limited and the main gates are organized stations, e.g. one can consider the airports.

Border survey is a sample survey of incoming (non-resident) and outgoing (residents) people with the primary objective of estimating, on a monthly basis, the cost of non-residents in Greece and residents abroad. The survey covers all outbound and inbound traffic of each type of border station (airports, ports, road, and rail border stations).

The data was collected in multiple files by using a web scraping method for downloading the corresponding content. A large number of files and the heterogeneity between them led to the conclusion that they had to be preceded by an initialization, be divided into small groups and then more efficiently processed.

The files were in following grouped in different categories, according to the following characteristics:

- content including data about arrivals and nights spent in Greece,
- qualitative characteristics of resident tourists and
- details about hotels, rooms for rent as well as tourist campsites.

Some combinations will examine specific groups regarding the gender and the age groupings or trips with 4 or more overnight stays in the country or even the trips with 4 or more overnight stays to relatives or friends.

[9]https://en.wikipedia.org/wiki/Linear_regression

There are a few different ways to write data to a Cassandra table including the CQL command “insert into” or the “copy” command which is useful for importing data from CSV files. One more option for loading data in a table is by using the stable loader tool that Cassandra provides for this reason. The “insert into” command is mainly utilized for writing single rows of data, which might come from a client application or a web application where users can enter data in a text field. Of course, it can be mainly used for manually entering data on the command line. Because of the complexity and the amount of our created data set, the copy command was used for importing data from .csv files to the table that we have created in Cassandra.

Once the abovementioned files were collected, a conversion into a different format, before the import into the database, was performed. The conversion was accomplished by using a custom file data processing script written in Python.

C. System Description

The data modeling procedures including the creation of objects, reading and writing data, can be accomplished in two ways. The first solution is, of course, the execution of CQL commands directly on the Cassandra CQL console (Cqlsh), which is the replacement of the old command line interface (Cassandra-cli). The second key is the use of a high-level Cassandra client for Java, Ruby, C#, Python, Perl, PHP C++ and other languages developed by third parties for interacting with the Cassandra binary protocol.

In our work, an additional solution is also provided, which is the use of the Cassandra-Spark connector. By using this connector, both platforms’ functionalities are extended. Furthermore, the limited capabilities of Cassandra in dealing with data can reach their full potential by using Spark’s flexible RDD API. The essential advantage of this corresponding integration is that a whole set of aggregated functions (like SUM, MAX, COUNT, etc.) can be used on top of CQL queries. In addition, one other important advantage is that more than one table can be joined by leveraging the transformation functions that the RDDs provide (map, join, union) [1]. The system overview is illustrated in Figure 3 including all major components and their implementation.

In the current performance, the system is based on a server running a single node Cassandra cluster and a single node Spark cluster. In addition, all the latest versions of the programs were installed. The specific kind of the applications are shown in Table I.

TABLE I
INSTALLED PROGRAMS VERSIONS

Application	Version
Ubuntu Server	16.04.3
Java Virtual Machine	8
Apache Spark	2.2.1
Apache Cassandra	3.11.1

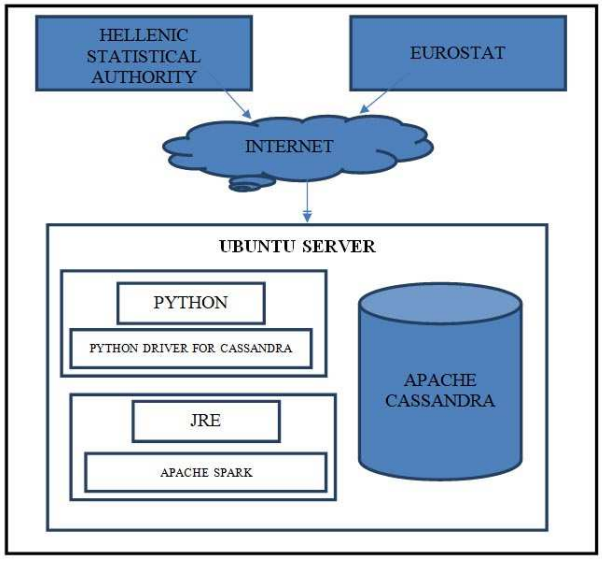

Fig. 3. The Overview of the System

V. EVALUATION

In supervised machine learning, the initial data is labeled and specific procedures are followed as a workflow before the evaluation of the results takes place. Namely, the typical procedure consists of the following six stages:

- **Data Collection:** The collected data should meet some validity requirements in order to provide reliable enough answers.
- **Data Preprocessing:** The quality of the data should be initially checked with the sampling method. In case of missing values, they should be filled, and the data should be finally scaled and normalized. The extraction process of the features should also be defined. In cases of massive and text-based datasets, tokenization and TF-IDF should be applied.
- **Data Transformation:** The algorithms can take as input particular formats of the data. The library MLlib of Spark accepts vectors (local, dense or sparse), labelled points as well as matrices (local, distributed, with a row, indexed row coordinate, and block).
- **Algorithms Application:** Since the suitable algorithms are verified for the problem, they should be applied in order to get the final results. Regardless of how many algorithms are to be applied, the final scores will be evaluated as the last step with the aim of selecting the most suitable one. Also, a combination of algorithms can be implemented in such a way that the best results are achieved.
- **Optimization Process:** The algorithm parameters might need a little fine tuning so as to produce better results. Testing of the parameters can take place during the training phase and can be also applied in the production phase.

- **Results Evaluation:** All the models should be graded in order to choose the best ones according to quality, scalability, performance and accuracy indicators; this stage takes place before proceeding to live data processing.

The new DataFrame based API for the library MLlib of Spark 2.0 is providing new and exciting possibilities. This higher level of programming abstraction is also an extension to the existing RDD API. RDDs are usually used to perform low-level transformations and actions over unstructured data. On the other hand, DataFrames are used to perform high-level expressions, to extract better data analytics and to execute SQL queries. The new API is the distributed collection of the objects in the form of rows and named columns, which is similar to tables in the RDBMS, but with much richer functionality. Therefore in our model, we have implemented a DataFrame for the input of our data and in following used its summary statistics to determine if data preprocessing is required. The output of df.describe() is shown in the following Table II.

TABLE II
RESULTS OF DF.DESCRIBE()

Summary	Arrivals
Count	10
Mean	6.82088256E7
Stddev	5,924,782.6,557,662,756
Min	57,796,551
Max	78,332,342

In our case, the minimum and maximum values of arrivals do not have a wide range of values, so there is no need of normalizing our data. Also, the values of the independent variables did not need any further adjustment, as they derived from the standard deviation value. Thus, we separated the features from the target variable and performed a linear regression.

In regression analysis, we have constructed a model to predict the value of a numerical variable, based on the value of other variables. We call the variable we want to predict, as "dependent variable" and the variables that are used to make the predictions are called "independent variables". In simple linear regression, we use a single numerical independent variable x for predicting the numerical dependent variable y [7].

Since the dependent variable y is not categorical, we did not use logistic regression; as a result we apply linear regression algorithms to the dataset and in the end evaluate the results. More specifically, we have noticed obvious correlations between the different variables. This type of linear relationship is needed between the independent (arrivals) and the dependent (nights spent) variables, and we are confident that our model is suitable for the corresponding data.

We have used the linear regression in order to explain the correlation existing between the tourist data and various independent variables. There is a capability of testing the different combinations of dependent and independent variables, but in the present paper, we chose to examine only the correlation between the tourist arrivals and the nights spent for the period $2006 - 2015$. Another possibility would be to test the correlation in different categories, like tourists originating from Greece and tourists originating from abroad or even for different Greek regions. The depended and independent variables used for the simple regression model are derived from the data of Table III.

TABLE III
ARRIVALS OF TOURISTS IN GREECE AND NIGHTS SPENT FOR PERIOD 2006-2015

Year	Arrivals	Nights spent
2006	13,982,021	57,796,551
2007	16,037,592	65,420,236
2008	16,013,569	65,624,563
2009	16,304,677	66,022,270
2010	16,241,395	66,800,371
2011	16,745,606	70,847,874
2012	14,790,696	64,384,415
2013	16,325,762	71,469,189
2014	17,743,493	75,390,445
2015	18,821,179	78,332,342

The new DataFrame we have implemented contains, apart from the labels and the features, a new prediction column as well. This particular column maps the plain old values and also extracts the labels. Hence, the results provide us the output of the prediction model as well as its known values.

As a result, with this way, we could quickly compare these outputs since they are printed side by side. The results of the predicted values, which are presented on Table IV, define a correlation between our actual observed values and the predictions; this ensures us that we can make adequate predictions using the previously constructed model.

TABLE IV
ACTUAL OBSERVED VALUES AND THE PREDICTIONS

Actual Observed Values	Predictions
49,761,406.03,433,327	57,796,551
68,286,817.5,132,859	66,022,270
56,211,365.53,231,404	64,384,415
79,762,756.4,566,769	75,390,445
88,358,336.94,518,083	78,332,342

For further evaluation, more accurate metrics are required. For this reason, we also compute the values of the coefficient and intercept. The coefficient metric measures the proportion of variation in dependent variable y that is explained by the variation in the independent variable x of our regression model [7].

By using the summary() function, detailed information about our model can be extracted. More specifically, this information contains the formula, the distribution of the residuals, as well as a piece of information about the overall performance of the R-squared.

TABLE V
RESULTS OF THE SUMMARY METHOD

Residuals
-1,437,498.8,308,113,962
-1,141,577.5,834,174,454
-1,853,799.3,254,712,373
1,290,642.4,587,702,155
2,280,811.655,619,465
861,421.6,253,103,614
RMSE
1,550,058.127,297
r2
0.899185

As depicted in Table V, the value of R-squared (i.e. coefficient of determination) is 0.899185 which means that almost 90% of the dependent variable (nights spent) is predictable by the independent variable (arrivals). In general, this metric shows how close to the fitted regression line our data are. The range of the values of R-squared will always be from 0 to 100%.

When the value of R-squared is 0, then the model explains none of the variability of the respond data around its mean. On the other hand, values near 100% describe the exact opposite and this aspect explains all the variability. In general, as the value of R-squared increases, then it means that our model fits into the data and in our case the percentage is 89%, which is a very high score. We also get information about the mean square error which is $1,550,058$ and constitutes an estimate of the observed nights spent. When an RMSE has a small value, then the predicted values are closer to the observed ones.

The residual or estimated error value is the difference between the values of the dependent and independent variables. Usually, with residual analysis, we can visually evaluate the four assumptions of regression and decide whether the model we have selected is appropriate. The R-squared value represents the amount of variation in y explained by variable x [7].

To even better evaluate our model, we should make some optimization in the process by tuning the parameters and utilizing again the testing. After the testing, we can decide which changes produced the better results. The initial values we used were the following:

$$lir = LinearRegression(maxIter = 10, \\ regParam = 0.3, elasticNetParam = 0.8)$$

Our findings indicate that after the optimization process, no significant improvement has been recorded. Our final opinion for our regression model was to keep the values we chose at the beginning. Of course with further research, our model could be further improved. Nevertheless, since we have decided that our model is appropriate for our needs, we can use it to make predictions about the nights spent by tourists. We must be aware that by using a regression model for prediction, the values entered in the previous code should comprise the relevant range of the independent variable. In the development of the regression model, all the values (i.e. from the lowest to the highest) are considered. The correct methodology dictates that in order to predict the value of y for a given value of x, we should extrapolate out of the range of x values. On the contrary, we should interpolate in the relevant range of the corresponding x values [7].

VI. CONCLUSIONS

In our proposed work we have presented a methodology for modeling Heterogeneous and Semi-Structured Information using NoSQL Databases. The data was collected from the websites of Hellenic Statistical Authority and Eurostat. Because of the heterogeneity between the selected files, they were divided into small sets and then grouped into different categories using custom python scripts. The result was an integration of Apache Spark with Apache Cassandra and the construction of a robust data analytics platform. Concretely, Apache Cassandra is considered as a NoSQL database for handling large amounts of data, thus capable of providing highly available service with no single point of failure. The tables created in Cassandra were designed according to the required queries and the database schema principles.

What is more, a table containing only the necessary data for our evaluation process was created, complying exactly with Cassandra result oriented model and acting as a pre-built result set. Since Cassandra is a column-oriented store which is a multidimensional map, the map data structure was used; this structure offers efficient key lookups as well as efficient scans. We applied the proposed data model to the corresponding dataset and the model was evaluated with the utilization of a machine learning algorithm (i.e. in a linear regression scenario) using Apache Spark and Apache MLlib. The results of our work have shown that additional testing should be performed regarding the validity of our model. In particular, it should be checked whether the algorithm has the same performance and behavior in one particular segment of data within a single set. Moreover, it should be checked whether the results are precise for a whole random dataset even if it contains completely different data. The proposed model is dynamic and customizable and can be used by other researchers on other scientific fields as well.

VII. FUTURE WORK

As far as future work is concerned, we plan to create a web-based platform for the registration, processing, and final analysis of the data. Once this particular model has been developed, the platform will be built with the use of a modern framework. Using a simple user interface (UI), any user will have the ability to retrieve data from sources such as ELSTAT and EUROSTAT with relative ease. The system will in following be able to suggest ways to proceed with the pre-processing stage. After the choice of the final format for the data transformation, the platform will provide the option for an algorithm to be applied by selecting from a list of

the eligible ones. The parameters will be dynamic, enabling easier tuning and testing of the results. Also, statistics will be presented based on specific metrics, such as quality, scalability, performance, and accuracy. At the end of the procedure and depending on the final score, the user will be able to choose the most appropriate algorithm from the chosen ones. The results can be stored in a separate table in the database for future study.

Although the evaluation of our proposed model has shown satisfactory results in terms of corresponding performance, additional research should be carried out in other areas as well. As abovementioned, the current methodology refers to tourism statistical elements for a single country, which, in our paper, is Greece. The statistical service in other countries could exploit more indicators so that the regression algorithm could be differentiated and in following the results could be evaluated in other way. Indeed, we therefore firmly believe that additional parameters should be used if we deal with larger quantity of data and perhaps a multi-regression model could be also applied to such heterogeneous data.

Finally, for the maximum exploitation of the benefits of the proposed model, a further modification could be recommended. That is the development of a clustered system as in our proposed work, the data were tested at a single node. In this case, the efficient management of larger scale data would be very promising. Not to mention that the research on such a distributed system can also bring valuable conclusions for future application development. Furthermore, one other aspect that can be taken into consideration concerns the potential modifications regarding the hardware specifications. As we are familiar with, the percentage of system performance is relative to the increase of hardware memory or possibly to other hardware modifications as well. Further research on the details of these conditions could yield promising and significant results.

REFERENCES

[1] C. Y. Akan. *Cassandra Data Modeling and Analysis*. Packt Publishing, 2014.

[2] A. Alexopoulos, A. Kanavos, K. Giotopoulos, A. Mohasseb, M. Bader-El-Den, and A. Tsakalidis. Incremental learning for large scale classification systems. In *Artificial Intelligence Applications and Innovations (AIAI)*, pages 112–122, 2018.

[3] P. Atzeni. Data modelling in the nosql world: A contradiction? In *17th International Conference on Computer Systems and Technologies (CompSysTech)*, pages 1–4, 2016.

[4] P. Atzeni, C. S. Jensen, G. Orsi, S. Ram, L. Tanca, and R. Torlone. The relational model is dead, SQL is dead, and I don't feel so good myself. *SIGMOD Record*, 42(2):64–68, 2013.

[5] A. Badia and D. Lemire. A call to arms: Revisiting database design. *SIGMOD Record*, 40(3):61–69, 2011.

[6] A. Baltas, A. Kanavos, and A. Tsakalidis. An apache spark implementation for sentiment analysis on twitter data. In *International Workshop on Algorithmic Aspects of Cloud Computing (ALGOCLOUD)*, pages 15–25, 2016.

[7] M. Berenson, D. Levine, K. A. Szabat, and T. C. Krehbiel. *Basic Business Statistics: Concepts and Applications*. Prentice Hall PTR, 7th edition, 1998.

[8] F. Bugiotti, L. Cabibbo, P. Atzeni, and R. Torlone. Database design for nosql systems. In *33rd International Conference on Conceptual Modeling (ER)*, pages 223–231, 2014.

[9] R. Cattell. Scalable SQL and nosql data stores. *SIGMOD Record*, 39(4):12–27, 2010.

[10] M. Chevalier, M. E. Malki, A. Kopliku, O. Teste, and R. Tournier. Implementation of multidimensional databases in column-oriented nosql systems. In *19th European Conference on Advances in Databases and Information Systems (ADBIS)*, pages 79–91, 2015.

[11] J. Han, E. Haihong, G. Le, and J. Du. Survey on nosql database. In *6th International Conference on Pervasive Computing and Applications (ICPCA)*, pages 363–366, 2011.

[12] A. Kanavos, N. Nodarakis, S. Sioutas, A. Tsakalidis, D. Tsolis, and G. Tzimas. Large scale implementations for twitter sentiment classification. *Algorithms*, 10(1):33, 2017.

[13] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia. *Learning Spark: Lightning-fast Big Data Analysis*. O'Reilly Media, Inc., 2015.

[14] R. Kienzler. *Mastering Apache Spark 2.x*. Packt Publishing, 2017.

[15] C. Li. Transforming relational database into hbase: A case study. In *IEEE International Conference on Software Engineering and Service Sciences (ICSESS)*, pages 683–687, 2010.

[16] X. Meng, J. K. Bradley, B. Yavuz, E. R. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. B. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241, 2016.

[17] A. B. M. Moniruzzaman and S. A. Hossain. Nosql database: New era of databases for big data analytics - classification, characteristics and comparison. *International Journal of Database Theory and Application*, 6(4):1–14, 2013.

[18] N. Nodarakis, S. Sioutas, A. Tsakalidis, and G. Tzimas. Large scale sentiment analysis on twitter with spark. In *EDBT/ICDT Workshops*, 2016.

[19] D. Pasqualin, G. Souza, E. L. Buratti, E. C. de Almeida, M. D. D. Fabro, and D. Weingaertner. A case study of the aggregation query model in read-mostly nosql document stores. In *20th International Database Engineering & Applications Symposium (IDEAS)*, pages 224–229, 2016.

[20] T. Rabl, M. Sadoghi, H. Jacobsen, S. Gómez-Villamor, V. Muntés-Mulero, and S. Mankowskii. Solving big data challenges for enterprise application performance management. *CoRR*, abs/1208.4167, 2012.

[21] S. Ries. *OCA Oracle Database 11g: Database Administrator I: A Real-World Certification Guide*. Packt Publishing, 2013.

[22] S. Ryza, U. Laserson, S. Owen, and J. Wills. *Advanced Analytics with Spark: Patterns for Learning from Data at Scale*. O'Reilly Media, Inc., 2017.

[23] A. Schram and K. M. Anderson. Mysql to nosql: Data modeling challenges in supporting scalability. In *Conference on Systems, Programming, and Applications: Software for Humanity (SPLASH)*, pages 191–202, 2012.

[24] D. Serrano, D. Han, and E. Stroulia. From relations to multi-dimensional maps: Towards an sql-to-hbase transformation methodology. In *8th IEEE International Conference on Cloud Computing (CLOUD)*, pages 81–89, 2015.

[25] J. Shi, Y. Qiu, U. F. Minhas, L. Jiao, C. Wang, B. Reinwald, and F. Özcan. Clash of the titans: Mapreduce vs. spark for large scale data analytics. *PVLDB*, 8(13):2110–2121, 2015.

[26] J. Shute, R. Vingralek, B. Samwel, B. Handy, C. Whipkey, E. Rollins, M. Oancea, K. Littlefield, D. Menestrina, S. Ellner, J. Cieslewicz, I. Rae, T. Stancescu, and H. Apte. F1: A distributed SQL database that scales. *PVLDB*, 6(11):1068–1079, 2013.

[27] S. Sioutas, P. Mylonas, A. Panaretos, P. Gerolymatos, D. Vogiatzis, E. Karavaras, T. Spitieris, and A. Kanavos. Survey of machine learning algorithms on spark over dht-based structures. In *Second International Workshop on Algorithmic Aspects of Cloud Computing (ALGOCLOUD)*, pages 146–156, 2016.

[28] M. Stonebraker. SQL databases v. nosql databases. *Commun. ACM*, 53(4):10–11, 2010.

[29] M. A. Vouk. Cloud computing - issues, research and implementations. *Journal of Computing and Information Technology (CIT)*, 16(4):235–246, 2008.

[30] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica. Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65, 2016.

[31] P. Zikopoulos, C. Eaton, et al. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 2011.